2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

Resource Constrained CVD Classification Using Single Lead ECG On Wearable and Implantable Devices

Arijit Ukil[1], Ishan Sahu[1], Angshul Majumdar[2], Sai Chander Racha[1], Gitesh Kulkarni[1], Anirban Dutta Choudhury[1], Sundeep Khandelwal[1], Avik Ghose[1], Arpan Pal[1]

[1]TATA Consultancy Services, India, [2] Indraprastha Institute of Information Technology, Delhi, India
e-mail[1]: (arijit.ukil, ishan.sahu, sai.racha, gitesh.k, anirban.duttachoudhury, sundeep.khandelwal, avik.ghose, arpan.pal)@tcs.com; email[2]: angshul@iiitd.ac.in

Abstract— **Electrocardiogram (ECG) is one of the fundamental markers to detect different cardiovascular diseases (CVDs). Owing to the widespread availability of ECG sensors (single lead) as well as smartwatches with ECG recording capability, ECG classification using wearable devices to detect different CVDs has become a basic requirement for a smart healthcare ecosystem. In this paper, we propose a novel method of model compression with robust detection capability for CVDs from ECG signals such that the sophisticated and effective baseline deep neural network model can be optimized for the resource constrained micro-controller platform suitable for wearable devices while minimizing the performance loss. We employ knowledge distillation-based model compression approach where the baseline (teacher) deep neural network model is compressed to a TinyML (student) model using piecewise linear approximation. Our proposed ECG TinyML has achieved ~156x compression factor to suit to the requirement of 100KB memory availability for model deployment on wearable devices. The proposed model requires ~5782 times (estimated) less computational load than state-of-the-art residual neural network (ResNet) model with negligible performance loss (less than 1% loss in test accuracy, test sensitivity, test precision and test F1-score). We further feel that the small footprint model size of ECG TinyML (62.3 KB) can be suitably deployed in implantable devices including implantable loop recorder (ILR).**

I. Introduction

With the advent of off-the-shelf sensor technologies coupled with advancement and rapid developments of Artificial Intelligence (AI) techniques, Cardiovascular Disease (CVD) detection using single lead ECG is becoming increasingly popular. Proposed techniques are proven to be helpful in diagnosing CVDs including transient infrequently arrhythmias especially Atrial Fibrillation (AF) and rhythm monitoring. Portable ECG devices currently offer an efficient screening option for AF; generating comparable performance to 24 hours Holter monitoring [1]. A study using Kardia band (single lead ECG) demonstrated moderate diagnostic accuracy when compared to 12-lead ECG analysis. The study also concluded that combining the automated device diagnosis with Electrophysiologists' (EP) interpretation of unclassified tracings yielded improved accuracy. Future improvements in automated algorithms were required with physicians' involvement when exploring the utility of these devices [2]. Such diagnostic inference on single lead ECG often requires sophisticated deep learning (DL) models. We find two critical problems of running such DL based ECG diagnostics natively on the typical resource-constrained wearables: 1. Depending on the number of layers, size of a DL model may become too high. In order to be wearable ready, the model size requires to be as small as possible (preferably sub-100 KB) 2. The associated battery drain which also in turn demands for smaller and less compute heavy models. In this paper, we propose a piecewise linear approximation of a ResNet [3] based ECG diagnostic inferencing model (ECG TinyML) that takes 156 times less memory than the original Resnet model. The proposed ECG TinyML is 5782 times less computationally intensive compared to the baseline ResNet model, with almost no compromise in classification performance. The final reduced model takes less than 70 KB of memory, making it suitable for embedding into cardiac implantable devices like ILR.

II. On The problem of ECG Classification Model for Wearable Devices

A. The Application Landscape

As illustrated in [4], cardiac rhythm monitoring and management devices have had a large proliferation over the past decade, and hence the on-device detection of cardiac rhythm anomalies is an important problem to solve. Further, [5] provides overview of methods and challenges of analyzing single lead ECG (e.g. KardiaMobile [6]) for clinical outcome, which proves that the problem is non-trivial. The problem is further amplified if we need to perform the detection on a resource constrained device like a wearable or an implantable device that have limited memory and battery power. [4] provides an analysis different cardiac rhythm management devices and associated therapies in Australian market. Though 75% of those devices are traditional pace makers, other devices capable of defibrillation and cardiac resynchronization have found their places in the list.

B. The Hardware Landscape

ILRs and wearable devices are generally composed of tiny microcontroller units (MCU) and specific sensors [7]. The block diagram of a typical hardware system is shown in Fig. 1. It consists of a main MCU which reads data from connected sensors. A separate MCU or System-On-Chip (SoC) does the communication with external world using Bluetooth Low Energy (BLE) or other low energy communication protocols. Such a hardware setup will have both costlier and faster volatile memory (RAM) and slower but cheaper non-volatile memory such as an external flash.

The main MCU in these devices is often severely resource constrained. They typically range from low-end 32-bit ARM

Cortex M0 to Cortex M7 with limited flash memory (from 32KB to 1MB) and RAM sizes ranging from 10KB to 256KB. Even if an additional flash memory is externally connected to the MCU to increase total memory size, such connection increases power consumption by many folds due to the increased I/O, thereby leading to reduced battery life. Also, slower memory such as flash demands more energy [8]. It was shown in [9] that 1mJ/KB is the average energy consumed by the whole system of the wearable in a sequential data write to local flash indicating a directly proportional (i.e. linear) relation between storage size and energy consumption. In a typical wearable with single sensor, the wearable storage accounts for almost 1/3rd energy [9]. Memory may be put on the same die as the core to reduce power consumption, but such on-die memory is pricier and hence difficult to integrate in low-end wearables.

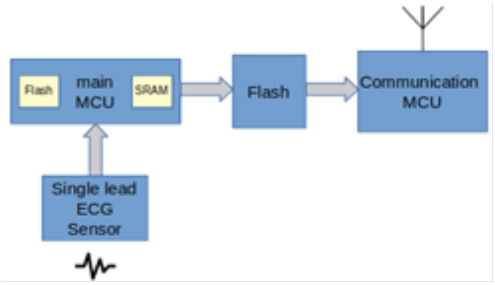

Figure 1. Single lead ECG analyis on MCU

In premium wearables and loop recorder that can afford higher amount of memory (Flash and RAM) in SoC, battery consumption still dominates the design decisions. This is because battery consumption gets greatly affected by the residence of the code (flash or RAM). For example, a code executed from the flash memory consumes 275µA/MHz as compared to when executed from the RAM (150µA/MHz) [10]. Communication takes up a significant energy. It is seen that just by sending feature extracts instead of sending raw data at 100Hz sampling of sensor, close to 20% of total energy was saved [11]. Memory as well as the energy cost of bigger models is one more important factor. All these factors above indicate that it is imperative to reduce the size of DL models to fit in the limited memory available in these devices and adhere to the power consumption requirement so that they can qualify as medical grade device. Authors in [12], provide the hardware landscape of such implantable or wearable devices and their functionality. The paper does not provide the hardware specifications for the devices, however, considering that they are battery powered and long running devices, we can consider the specifications of a low-power microcontroller platform to exist within the device.

C. The Proposed Solution Overview

Let us consider that certain pre-trained DNN model $\mathcal{D}$ achieves performance metric (say, in terms of accuracy, which can also be measured in terms of sensitivity, F1-score, etc.) ρ and the trained model size is $\mathcal{M}$KB for an ECG classification task $\mathcal{E}$. Let us further consider the following: target micro-controller of wearable device is having memory budget $\mathcal{M}^{compr}$KB and the allowed performance penalty is $\delta, \delta \ll 1$. Let the performance metric of the micro-controller deployed trained model be ρ' and $\frac{\rho-\rho^{compr}}{\rho} \leq \delta$.

In a likely scenario, we get $\mathcal{M} \gg \mathcal{M}^{compr}$. In order to achieve a deployable trained model, we need to have at least $\frac{\mathcal{M}}{\mathcal{M}^{compr}}=\alpha$ compression gain while $\frac{\rho-\rho^{compr}}{\rho} \leq \delta$. Classically, such model compression need can be attempted using quantization and pruning approaches [7]. However, we find that typical wearable micro-controller memory budget is less than 100KB ($\mathcal{M}^{compr} < 100$) and a novel model compression approach is required. ECG TinyML is such a novel model compression method that fulfills the criteria for ECG classification at wearable devices. Additionally, and importantly, we need to significantly reduce the run time or equivalently computational load (less computational load results in less power consumption) of the deployable compressed model, which can be estimated from the number of floating point operations (FLOPs) of the base line model ($\mathcal{F}$) and that of compressed model ($\mathcal{F}^{compr}$), i.e. $\frac{\mathcal{F}^{compr}}{\mathcal{F}} < \theta$, where, $\theta \ll 1$.

III. TinyML for ECG Classification- Proposed Model Compression Algorithm

Larger and complex DNN models are the current state-of-the-art for solving different classification problems, including ECG classification tasks [3]. Such networks often consist of many million parameters and the memory requirement is too demanding which might not be supported by tiny edge devices for in-situ inference purpose. In order to reduce the complexity of the baseline DNNs, model quantization and pruning are the widely-used approach [11, 13].

A. Quantization and Pruning Based Deep Network Compression- Classical Approach

Conventionally, pruning and quantization are the widely-used model compression techniques. Pruning helps in removing unnecessary connections of the weight tensor of the baseline DNN to reduce the model storage memory size. Quantization on the other hand is reducing the precision of the datatype into lower bits such that the consumption of memory on the device is less compared to the original model. In a typical deployment scenario, baseline model is pruned using TensorFlow (TF) model optimization and quantization is done to integer data type precision to reduce the memory and computational overload [11, 16]. We observe that pruning and quantization methods are not sufficient to reduce the baseline DNN for MCU-based ECG analytics.

B. Piecewise Linear Approximation for Deep Network Compression as ECG TinyML- Our Approach

In this paper, we employ a novel Knowledge Distillation (KD) method to uniquely compress a baseline DNN model to achieve significant compress gain while maintaining similar performance metric (in terms of test accuracy, test sensitivity, test precision, test F1-score metrics) [13]. KD approach enables effective transfer of the knowledge from a large pre-trained DNN model (teacher model) to a compact model (student model). Contrary to conventional KD method, where homogeneity in the teacher-student architecture is maintained (i.e. student model is smaller DNN), we propose heterogeneity in the KD process, where student model architecture is a shallow machine learning algorithm.

Let us consider a given ECG dataset which consists of signals $\mathcal{S}$, and their associate label vector $\mathcal{L}$. Let $\mathcal{g}(.)$ denote all the operations performed till the penultimate layer, i.e, the layer just before the last softmax layer of the trained DNN model. Then we have, $\rho_{out} = \mathcal{g}(\mathcal{S})$, and predicted label $\mathcal{L}' = softmax(\rho_{out})$.

We try to estimate $\mathcal{g}(.)$ and then use the last layer of the trained DNN to generate the class labels. This gives us an alternative to directly learning the relationship between the input signal and hard labels, which is a more difficult problem. As the relationships learnt by DNNs are usually complex in nature, we take a piecewise approach: divide the input space into smaller pieces and then approximate the input $\mathcal{S}$ – output ρ_{out} relationships for each of them using separate linear models. The method is described below.

Algorithm: Piecewise linear approximation for baseline DNN model Compression- ECG TinyML (proposed approach)

Producing the compressed model from baseline DNN:

Input: Training ECG data- $\mathcal{S}$, trained DNN model- $\mathcal{D}$.
Output: Compressed model suitable for deployment on resource constrained devices: $\Pi^{ECG\ TinyML}$.
Procedure:
Step 1: We cluster (using *k*-means algorithm) $\mathcal{S}$ into k distinct groups and learn the set of cluster centroids $\mathcal{C}$. These clusters provide us the pieces of the input space of $\mathcal{S}$. The optimal number of clusters for this dataset = 3. We explore number of clusters in the range 1 to 5. Based on the 5-fold cross validation performance over the training data, we select the optimal number of pieces in which the input space is divided.
Step 2: The set of pairs $\{(\mathcal{S}, \rho_{out})\}$, is generated using $\mathcal{D}$, where $\rho_{out} = \mathcal{g}(\mathcal{S})$.
Step 3: These pairs, $\{(\mathcal{S}, \rho_{out})\}$, are associated with the corresponding cluster $\mathcal{R}$ of $\mathcal{S}$.
Step 4: For each element of $\mathcal{R}$, we learn a new linear approximation model by solving the linear least squares problem with regularization.

$$argmin_{W_k} \left\| \rho_{k,out} - W_k \mathcal{S}_k \right\| + \lambda \|\mathcal{S}_k\|$$

$$W_k^T = (\mathcal{S}_k \mathcal{S}_k^T + \lambda I)^{-1} \mathcal{S}_k \rho_{k,out}^T$$

where, W_k represents the weights of the different linear models and λ is the regularization parameter.
Step 5: k, the number of clusters or pieces is tuned by searching over different values to achieve closest approximation of DNN.
Step 6: We get the compressed model, Π* which consists of cluster centroids, set of linear models for each of the clusters and the last layer weights of the DNN model ($\mathcal{D}$).
Step 7: Π* is further quantized to $\Pi^{ECG\ TinyML}$ for ensuring model compactness (Please refer Section IIIA).

ECG signal classification using $\Pi^{ECG\ TinyML}$:

Input: Test ECG signal: $\mathcal{S}_{test}$
Output: Label denoting health condition
Procedure:
Step 1: First the cluster membership of $\mathcal{S}_{test}$ is ascertained by determining the nearest cluster centroid present in $\Pi^{ECG\ TinyML}$.
Step 2: Using the linear regression model associated with the identified cluster, ρ'_{out} is computed.
Step 3: Finally, after multiplication of ρ'_{out} with the last layer weights, we get the predicted health condition.

IV. Experimental Results

In order to establish the efficacy of our proposed method, we experiment with publicly available ECG dataset [14][15] and use state-of-the-art classification model ResNet [3].

A. Dataset Description

ECG dataset is extracted from the BIDMC Congestive Heart Failure Database [15]. The original data is sampled at 250 Hz and pre-processed in two steps: (1) extract each heartbeat, (2) make each heartbeat equal length using interpolation with time series length equals to 140 time steps [14]. There are 5 different classes: Normal, R-on-T Premature Ventricular Contraction, Supraventricular Premature, Premature Ventricular Contraction, and Unclassifiable Beat. The number of training and testing instances are 500 and 4500 respectively. We have used 5-fold cross-validation on training data (500 instances) is used to tune the parameters. Since, it is 5-fold we have 80% train and 20% validation for each iteration.

B. Experimental Procedure

The ResNet based DNN model is considered as the baseline model for our ECG analytics application, which is a state-of-the-art algorithm for time series classification tasks including ECG classification [3]. The baseline model is trained on a GPU (Nvidia Quadro P5000 GPU with 16 GB GPU RAM) using the given training ECG data to get the pre-trained baseline DNN model $\boldsymbol{\mathcal{D}}$. We further compute the performance metrics of the baseline DNN model over the testing dataset. The pre-trained model is 9.71 MB ($\mathcal{M}$) in size. The MCU (e.g. STM 32 MCU) for edge analytics demands for sub-100 KB ($\mathcal{M}^{compr}$) model memory size. Thus, the model compression gain $\alpha = \frac{\mathcal{M}}{\mathcal{M}^{compr}} = \frac{9710}{100} = 97.1$ and we set the allowed performance penalty is $\delta = 1\%$.

In our method, there are two control parameters: k, the number of pieces or clusters and λ, regularization parameter for each of the respective linear models. In order to find the optimal model parameter, we tune them using 5-fold cross validation on the training data itself. For k, we consider the range 1 to 5. And for λ, search was done over the values ranging from 0.01 to 1000. The resultant compressed model is then deployed on the microcontroller and tested using the test data.

B. Results

The classification inference performance of our solution along with the comparison with baseline DNN model is detailed in Table II. We measure the weighted performance metrics including test accuracy, test sensitivity, test precision and test F1-score for the comparative study. In Table I, we depict the limitation of pruning and quantization approach, where the compressed model size is achieved to 801 KB, which does not fit for sub-100 KB model footprint. The typical model construction workflow for edge deployment (where we need to deploy Tensorflow (TF) Lite model [17]) is as follows: Baseline TF model → TF Lite model → Compressed TF Lite model [16]. The model deployment workflow for ECG TinyML is as flows: Baseline TF model → Piecewise Linear Approximation (PLA) model → PLA with quantization in TF

Lite format. We have applied integer with float fallback with INT8 quantization.

TABLE I. MODEL COMPRESSION BY CLASSICAL PRUNING AND QUANTIZATION APPROACHES

Method	Method Performance				
	Test accuracy	*Test sensitivity*	*Test precision*	*Test F1-score*	*Model size*
Baseline DNN (ResNet) [28]	0.931	0.931	0.924	0.926	9.71 MB
Baseline model TF Lite format	0.931	0.931	0.924	0.926	3.13 MB
Pruned and quantized TF Lite	0.929	0.931	0.927	0.928	0.801 MB

TABLE II. MODEL COMPRESSION BY OUR PROPOSED ECG TINYML METHOD

Method	Method Performance				
	Test accuracy	*Test sensitivity*	*Test precision*	*Test F1-score*	*Model size*
Baseline ResNet [3]	0.931	0.931	0.924	0.926	9.71 MB
Piecewise Linear Approximation (PLA)	0.938	0.938	0.926	0.929	0.443 MB
PLA-quantized TF Lite (ECG TinyML)	**0.937**	**0.937**	**0.925**	**0.928**	**0.062 MB**

We like to mention that the number of parameters of baseline ResNet model is ResNet 804,169, whereas there are 55710 parameters in our proposed compressed PLA model. Another important aspect of this experimental study is to demonstrate the low computational load in terms of FLOPs in ECG TinyML than the baseline ResNet model as shown in Table III. We also notice the unchanged computational load of pruned and quantized model (pruning and quantization algorithms are designed to reduce the model memory size, not to reduce the computational load like the number of matrix multiplications in DNN models).

TABLE III. COMPUTIONAL LOAD (IN TERMS OF FLOPS) COMPARISON

Method	*FLOPs*
Baseline DNN (ResNet) [28]	222908190
Baseline model pruned and quantized TF Lite	222908190
PLA- quantized TF Lite (ECG TinyML)	38554

C. Result Summary and Observation

Our experimental results reveal that the proposed ECG TinyML demonstrates high model compression gain ($\alpha = 156$), insignificant performance penalty ($\delta \cong 0$) and substantially less computational load for effective run time execution ($\theta = 1.72 \times 10^{-4}$) and consequently, higher energy saving. Hence, ECG TinyML is well-suited for edge deployment. Subsequently, the very tiny model foot print of 62.3 KB and low FLOPs of 38554 pave us to explore the feasibility of ECG TinyML deployment in cardiac implantable devices like ILR.

V. CONCLUSION

The primary objective of ECG tinyML is to establish the feasibility of a transformation process for deploying sophisticated, complex and highly computationally intensive DNNs to the microcontroller based wearable ECG devices under the constraints of tinyML scenario. We intend that ECG tinyML will pave path towards automated detection of CVDs and act as a pivotal component for building early warning systems. We envisage our future work to encompass larger set of ECG classification tasks including different arrhythmias for edge deployment as well as to explore the likelihood of deploying in array of implantable devices including ILRs.

REFERENCES

[1] S. Ramkumar, N. Nerlekar, D. D'Souza D, et al. "Atrial Fibrillation Detection Using Single Lead Portable Electrocardiographic Monitoring: A Systematic Review and Meta-Analysis," BMJ Open 2018.

[2] K. Rajakariar, et al. "Accuracy of A Smartwatch Based Single-Lead Electrocardiogram Device in Detection of Atrial Fibrillation," Heart 2020.

[3] A. Ukil, S. Bandyopadhyay, and A. Pal, "Dyreg-fresnet: Unsupervised feature space amplified dynamic regularized residual network for time series classification," IEEE IJCNN, 2019.

[4] I. Stevenson, A. Voskoboinik, "Cardiac Rhythm Management Devices," Australian Journal of General Practice, 2018.

[5] S.M. Mathews, C. Kambhamettu, and K.E. Barner. "A Novel Application of Deep Learning for Single-Lead ECG Classification," Computers in Biology and Medicine, 2018.

[6] https://clinicians.alivecor.com/our-devices/.

[7] A. Srinivasulu and N. Sriraam, "An Engineering Perspective of External Cardiac Loop Recorder: A Systematic Review," Journal of Medical Engineering, 2016.

[8] H. Kim, N. Agrawal, C. Ungureanu, "Revisiting Storage for Smartphones," ACM Transactions on Storage, Dec 2012.

[9] J. Huang, A. Badam, R.Chandra, and Edmund B. Nightingale, "WearDrive: Fast and Energy-Efficient Storage for Wearables," Usenix Annual Technical Conference, 2015.

[10] L. Yankov, "Multiprotocol Bluetooth® low energy/2.4 GHz RF System on Chip, Product Specification v3.1, and Power and execution simulator for wearable devices," nRF51822.

[11] F. Grützmacher, et al. "Towards Energy Efficient Sensor Nodes for Online Activity Recognition," ICPEC, 2018.

[12] URL:https://www.researchandmarkets.com/reports/4871577/cardiac-rhythm-management-crm-devices-and?utm_source=dynamic&utm_medium=BW&utm_code=tz96jt&utm_campaign=1331634+-+Global+Cardiac+Rhythm+Management+(CRM)+Devices+and+Equipment+Market+Report+2020&utm_exec=chdo54bwd last visited on 31st March 2021.

[13] I. Sahu, A. Pal, A. Ukil, and A. Majumdar "Compressing Deep Neural Network: A Black-Box System Identification Approach" International Joint Conference on Neural Networks (IJCNN), 2021.

[14] Y. Chen, et al. "A General Framework for Never-Ending Learning from Time Series Streams," DMKD, pp. 1622-1664, 2014.

[15] A.L. Goldberger AL, et al. "Physiobank, Physiotoolkit, and Physionet: Components of A New Research Resource for Complex Physiologic Signals," Circulation, 2000.

[16] https://www.tensorflow.org/model_optimization.

[17] https://www.tensorflow.org/lite.